A Clean-Slate Look at Disk Scrubbing

Alina Oprea
RSA Laboratories
Cambridge, MA
aoprea@rsa.com

Ari Juels
RSA Laboratories
Cambridge, MA
ajuels@rsa.com

Abstract

A number of techniques have been proposed to reduce the risk of data loss in hard-drives, from redundant disks (e.g., RAID systems) to error coding within individual drives. Disk scrubbing is a background process that reads disks during idle periods to detect irremediable read errors in infrequently accessed sectors. Timely detection of such latent sector errors (LSEs) is important to reduce data loss.

In this paper, we take a clean-slate look at disk scrubbing. We present the first formal definition in the literature of a scrubbing algorithm, and translate recent empirical results on LSE distributions into new scrubbing principles. We introduce a new simulation model for LSE incidence in disks that allows us to optimize our proposed scrubbing techniques and demonstrate the significant benefits of intelligent scrubbing to drive reliability. We show how optimal scrubbing strategies depend on disk characteristics (e.g., the BER rate), as well as disk workloads.

1 Introduction

With the unremitting growth of digital information in the world, there is an ever increasing reliance on hard drives for critical data storage. Hard drives serve not only as primary storage devices, but due to their growing capacity and dropping prices, they are now an attractive building block for a range of storage systems, including large-scale secondary systems (e.g., archival or backup systems). In these environments, their reliability becomes significant and needs to be quantified, as some of these systems demand strict and high availability guarantees.

A significant body of research focuses on designing reliable storage systems by adding redundant disks. RAID systems enhance reliability by storing parity blocks in redundant arrays. Most systems today employ RAID-5 or RAID-6 mechanisms that are resilient to one or two simultaneous disk failures, respectively. Data loss in RAID is amplified by *latent sector errors* (LSEs), sector errors in drives that are not detected when they occur, but only when the disk area is accessed in the normal course of use. In RAID-5, a disk failure coupled with only one latent error on another disk induces data loss.

To increase the reliability of both single drives and RAID systems, researchers have studied techniques such as *intra-disk redundancy* [5] or *disk scrubbing* [15]. Intra-disk redundancy applies an erasure code over a subset (segment) of consecutive sectors in the drive and stores the parity blocks in the same disk. It protects against a small number of LSEs in each segment, depending on the parameters of the erasure code.

Disk scrubbing is a background process that reads disk sectors during idle periods, with the goal of detecting latent sector errors in infrequently accessed blocks. Most existing systems perform *sequential* disk scrubbing, meaning that they access disk sectors by increasing logical block address, and use a scrubbing rate that is constant or dependent on the amount of disk idle time. Mi et al. [9], for instance, suggest that disk scrubbing should be scheduled whenever the disk is idle in order to maximize scrubbing rates. A notable exception is the work of Schwarz et al. [15], which considers alternative scrubbing strategies with varying rates; the goal is to minimize disk power-on time in large archival systems whose disks are generally powered off.

In this paper, we define the first formal model for scrubbing strategies, along with a performance metric for the single-drive setting. Through a simulation model, we empirically search the space of scrubbing strategies and find optimal points in this space. We translate new results in the literature on the distribution of LSEs in hard drives

[2] into new scrubbing principles. The main message of the paper is that by exploiting a richer design space for scrubbing strategies, we can design better algorithms that significantly improve current technologies. We have to note, though, that our results are highly sensitive to some disk parameters that are not always made public by disk manufacturers. We hope that this paper will open up a new line of research that will further refine our results as more accurate disk failure data becomes available to the community.

In more detail, our main technical contributions are:

Formal model for scrubbing strategies We give the first formal model for scrubbing strategies that considers a number of disk parameters (e.g., disk age, disk model, disk failure rates), as well as history of disk usage. We view a scrubbing strategy as a function which, given information about a drive, outputs the set of sectors to be scrubbed in the next time interval.

The metrics most commonly used for hard drive reliability are MTTF (Mean Time To Failure) for single drives, and MTTDL (Mean Time To Data Loss) for a RAID system. For single drive reliability, MTTF measures the disk lifetime before total failure, and does not give a measure of its resilience to LSEs. MTTDL is a systemic measure, and not applicable to the study of errors in a single drive. Thus we define a new metric for hard drives called MLET ("Mean Latent Error Time"). MLET captures the percentage of time in which the disk is susceptible to data loss due to an LSE (and can serve as a basis for determining MTTDL). We define an optimal scrubbing strategy for a drive to be one that minimizes our new MLET metric.

Latent-sector error model Based on the results presented by Bairavasundaram et al. [2], and known results about usage-related LSEs [6], we propose a simple model for LSE development. Our model considers both age-related and usage-related LSEs, and captures their *spatial and temporal locality*. Since we do not have complete information about LSE distribution from the academic literature, we derive additional assumptions to generate a complete LSE model. We show that our model accurately reflects the field data presented by Bairavasundaram et al. We believe that our model is of general interest in the study of LSEs, as it provides a simplified and efficient tool for experimentation.

Find optimal strategy through simulation Guided by new empirical results on LSE distributions in the literature, we identify new scrubbing principles for single disks, summarized in Table 1. These principles suggest several new dimensions in the formulation of scrubbing strategies (e.g., variable scrubbing rates) and lead us to a newly enriched design space. Using a simulation based on our proposed LSE model, we search this design space for MLET-optimal scrubbing strategies. We find an optimal scrubbing strategy which, compared with straightforward sequential scrubbing, improves on the MLET metric by an order of magnitude.

Organization We review related work in Section 2. We create a model for the distribution of LSEs using the study of Bairavasundaram et al. [2] and additional assumptions, and validate this model against the study's empirical data in Section 3. We define scrubbing strategies formally, introduce our new design dimensions, and formulate our search space for scrubbing strategies in Section 4. We describe our simulation model and present our results on simulation-optimized scrubbing strategies in Section 5. We conclude in Section 6.

2 Related Work

Several recently published papers have shifted the storage community's perspective on disk failures in the real world. Schroeder and Gibson [14] show that annual disk failure rates are higher than those published by manufacturers, and determine that disks do not exhibit exponential times between failures (as commonly believed). Instead, time between failures is modeled more accurately by a Weibull distribution. Pinheiro et al. [11] offer statistics on disk survival rates conditioned on various SMART parameters. The first study on latent sector errors (LSEs) for field data is that of Bairavasundaram et al. [2]. They show that LSE rates increase linearly with disk age, and that LSEs are highly correlated, exhibiting both spatial and temporal locality.

Disk scrubbing is a well known technique used extensively to detect latent sector errors early. Most existing systems use a sequential scrubbing strategy in which sectors are read from disk in increasing order of their logical address. In the academic literature, more sophisticated scrubbing strategies have been proposed by Schwartz et al. [15] in the context of large archival storage systems. In such systems, one goal is to keep the disk powered down as much as possible, and minimize the number of power ups. Their opportunistic strategy piggybacks on normal read accesses—scrubbing when a disk is powered up for another operation. They also propose a simple, three-state Markov model that captures disk degradation due to scrubbing. Within this analytic model, they

Facts about LSE distribution	Corresponding proposed scrubbing principles
1. LSE rate is low in the first 60 days of operation	1. Keep scrubbing rate low during the first 60 days of operation
2. After 60 days, LSE rate is higher, but fairly constant before the first LSE develops	2. After 60 days, increase scrubbing rate and keep it constant before detecting a first LSE
3. LSEs exhibit temporal locality	3. Increase scrubbing rate after LSE detection
4. LSEs exhibit spatial locality	4. Staggered scrubbing (defined in Section 4.2) is superior to sequential or randomized scrubbing
5. LSEs develop as a function of disk usage	5. Scrubbing is not free: limit scrubbing rate to avoid collateral LSEs

Table 1: Translation of results on LSEs in the literature into scrubbing principles

calculate the optimal scrubbing rate.

To the best of our knowledge, our work provides the first general formalization of scrubbing strategies for hard drives and optimizes such strategies over a large search space. In contrast to Schwartz et al., we are interested in enterprise disks that are powered up most of the time, and we do not consider the power-up effect on reliability. Interestingly, we observe the adverse effect of aggressive scrubbing, much like Schwartz et al. While in [15], aggressive scrubbing detrimentally increases the number of disk power ups, in our system aggressive scrubbing triggers LSEs by increasing disk usage. Through our newly defined MLET metric, we are able to capture the effect of usage errors for drive reliability. We thus dispute the common belief that scrubbing is most effective at maximum capacity.

A number of research papers examine the effect of scrubbing and LSEs on RAID reliability. In his Ph.D. thesis [8], Kari developed the first Markov model for RAID reliability that considers LSEs (in addition to total disk failures). He obtained theoretical equations for MTTDL (the RAID reliability metric defined by Patterson et al. [10]), assuming that the distribution of LSEs is exponential. More recently, Elerath and Pecht [6] propose a 5-state simulation model for RAID-5, in which both the disk failure and LSE distributions are modeled by a Weibull probability density function.

Baker et al. [3] provide a reliability model for two-way mirroring in the context of long-term archival storage. In their Markov model, they consider exponentially distributed LSEs and their spatial and temporal correlation, which they model via an increased rate in their exponential distribution. They also show that scrubbing at a constant rate (every two weeks) reduces MTTDL.

Beyond scrubbing, there exist other single-disk techniques to protect against LSEs. Intra-disk redundancy schemes (IDR) [5] encode additional redundancy *within the disk itself* in the form of erasure codes. Dholakia et al. [5] propose encoding consecutive disk sectors under a custom-crafted XOR erasure code. Iliadis et al. [7] compare disk scrubbing and IDR with respect to RAID reliability. Mi et al. [9] consider the problem of scheduling background activities, including scrubbing and IDR, to increase the MTTDL metric for RAID. They show that combining scrubbing and IDR greatly improves RAID reliability.

3 Modeling the Distribution of Latent Sector Errors

We model the distribution of latent sector errors (LSEs) using the data presented in the recent NetApp study of Bairavasundaram et al. [2]. The NetApp study is the only published academic paper that gives a substantial characterization of LSE development. That said, the paper does not contain or reference detailed data: The LSE-development data sets on which the paper is based are proprietary, and have not been publicly released. Given these facts, our only choice to derive a meaningful LSE model was to reverse engineer some of the graphs presented in the NetApp paper. We make additional assumptions about LSE development as needed to generate a complete LSE model. We validate our LSE model against the graphs provided by the NetApp paper, but, of course, thorough validation of the model requires access to real data.

3.1 Results from NetApp study

The NetApp study [2] presents results on the LSE distribution of 1.53 million disks from various models and manufacturers over a 24-month period. The disks are divided into two classes: nearline and enterprise. In our work here, though, we restrict our study to enterprise disks. The main findings of the NetApp study on enterprise disks are summarized below:

1. LSEs develop at a fairly constant rate in the first two years of a drive's age. An exception are the first two months; these exhibit a slightly lower LSE rate. The fraction of disks developing at least one LSE is highly variable for different disk models, ranging at the end of the 24-month study from 1% to 4%.

2. LSEs exhibit *spatial locality* at the logical address level, as shown by two graphs in the paper. Figure 5 from the NetApp study shows the probability of another error within a given radius of an existing LSE. For most disk models, the probability of another latent error within 10MB of an existing error is 0.5. Figure 6 from the NetApp study shows the average number of errors within a given radius of an existing error. While both graphs provide some information about how LSEs are clustered together, the NetApp study does not provide full details about the exact probability distribution function of LSE locations in disks.

3. LSEs exhibit *temporal locality*. More than 80% of errors arrive at an interval of less than an hour from previous errors. Figure 7 in [2] shows that the inter-arrival time distribution has very long tails.

4. As shown in Figure 8 of [2], most additional errors occur in the first month after the first LSE, and the probability of developing these errors decays exponentially over time. For instance, the probability of a disk developing 1, 10, and 50 additional errors in the first month is 0.6, 0.25 and 0.1, respectively.

3.2 Latent sector error model

The NetApp study shows how latent errors develop in disks as a function of disk age. We call such errors *age errors*. Additionally, latent errors develop due to disk usage or disk wear-out. A hard-drive metric that captures usage is the *byte-error rate* (BER). While there is no consensus in the literature on the interpretation of this metric [4], we assume that both reads and writes contribute to development of usage errors, albeit with different weights. In our disk model, we vary the BER metric between 10^{-15} and 10^{-13} (to capture disks with various characteristics), and we define a read/write weight for each disk, denoted RW_Weight (to characterize the relative contribution of read and write operations to disk wear-out). We refer to the errors that develop due to disk wear-out as *usage errors*.

There is no explicit information in the academic literature about the exact distribution of usage-related LSEs. Since it is very likely that during the 24-month NetApp study at least several usage-related LSEs developed, we make the assumption that usage-related LSEs follow a spatial and temporal distribution similar to age errors.

The NetApp study shows that LSEs are clustered both spatially and temporally. We further categorize age and usage LSEs into two types of errors. The first type is that of *triggering errors*. We define a triggering error to be either the first age-related error in a drive, or the first usage-related error that develops after a specified amount of data has been accessed (counting from the time the previous usage-related error developed). A triggering error induces a cluster of additional errors, called *triggered errors*. These errors develop in a short interval of time after the corresponding triggering error, and are clustered spatially on disk closely to the triggering error.

Before giving full details on our LSE model, let us start with some intuition on modeling the spatial and temporal distribution of LSEs.

Modeling spatial distribution on disk As the NetApp study observes, most LSEs are clustered at radii of around 10-100MB. We define the *centroid* of a cluster to be the median error in the cluster with respect to block logical addresses. In our simulation model in Section 5, we need to generate errors in increasing order of occurrence time. For convenience in that model, we assume that the triggering error (i.e., the first error in a cluster) is also the cluster centroid. Since the NetApp study does not provide the exact location on disk of error clusters (but only error relative distance), we assume that the centroid location is uniformly distributed across all disk sectors. We model the triggered errors as being clustered around the centroid with radii determined from the distribution given in Figure 5 of [2]. In Section 3.3, we regenerate the graphs presenting spatial locality of LSEs in the NetApp study using our LSE model, in order to validate our simplifying assumptions.

Modeling temporal distribution We model the time at which a triggering error develops after the data in the NetApp study. Figure 1 in [2] gives the probability that a disk develops an age error in its first 24 months in the field; the results are presented at the granularity of six months. Combined with the results from Figure 10 in [2], we infer that the disk error rate is lower in the first 60 days of disk operation, and fairly constant after that. In our simulation model, we work at the temporal granularity of one hour. Without finer granularity on how triggering age errors develop temporally, we assume that the time a disk develops its first LSE error is uniformly distributed within the month in which the triggering error arises.

The time a usage error develops is determined by the disk BER metric, which we vary between 10^{-15} and 10^{-13}. We assume that usage error development follows a normal distribution with mean $1/\text{BER}$. A usage error is triggered once the number of bytes accessed (due to both normal disk workloads and the scrubbing process) weighted by RW_Weight, exceeds on average $1/\text{BER}$.

Once the occurrence time of the centroid is determined, we generate the number of additional errors in the disk based on the graph from Figure 8 in [2]. Figure 8 gives the probability of a disk developing up to 50 errors after a first LSE. The NetApp study does not provide a maximum limit on the number of LSEs in a disk, but it states that about 80% of disks develop less than 50 errors. We set the maximum number of LSEs in the disk to 100. The inter-arrival time for each triggered error is modeled with the distribution from Figure 7 in [2].

To generate the distributions from Figures 1, 5 and 7 in the NetApp paper we used piecewise uniform distributions with points given by those graphs. For Figure 8, we used curve fitting in Mathematica.

We summarize the assumptions made in generating our LSE model in Table 2.

1. Age errors form a single cluster on disk.
2. Usage error clusters develop due to both reads and writes, albeit with different weights.
3. Usage error clusters follow spatial and temporal correlations similar to those exhibited by age errors.
4. Development of a new triggering usage error follows a normal distribution with mean $1/\text{BER}$ and small deviation.
5. The triggering error of an error cluster is the cluster centroid.
6. Triggered errors developing closely in time are clustered around the centroid.
7. Cluster centroids are uniformly distributed on disk.
8. The time a triggering error develops in a month is uniformly distributed within the month.

Table 2: Assumptions for generating LSE model.

Formally, we define an LSE model as a probability distribution function $\mathcal{P}_{\mathsf{LSE}}$. First, let us define a bit vector E_t over all sectors in the disk, such that $E_t(s) = 1$ if sector s has developed a latent sector error at time t and $E_t(s) = 0$, otherwise. Taking as input time t, sector s, the cumulative write and read usage up to time t in bytes, denoted W_t and R_t, respectively, and the history of latent error development $E_1, \ldots, E_{t-1}$, $\mathcal{P}_{\mathsf{LSE}}(t, s, W_t, R_t, E_1, \ldots, E_{t-1})$ is the probability that sector s develops a latent sector error at time t. Let us denote the space of all LSE models as $\mathcal{L}$.

We give now full details on our LSE model.

1. **Modeling triggering age LSE.** Using Figures 1 and 10 from [2], we determine the probability that a disk develops an age error in each month of its first 24 months in the field. If a disk develops a triggering error in month $0 \leq m \leq 23$, then the exact occurrence time in hours is uniformly generated in the month, according to the distribution $U(720 * m, 720 * (m + 1) - 1)$. (Here $U(a, b)$ is the uniform distribution on $[a, b]$.)

2. **Modeling triggering usage LSE.** We fix the BER metric for a disk to a value in the set $\{10^{-15}, 10^{-14.5}, 10^{-14}, 10^{-13.5}, 10^{-13}\}$. Once the BER metric is fixed (e.g., 10^{-14}), a usage error is developed when Bytes_Written + Bytes_Read/RW_Weight $>=$ $1/\text{BER}$. If we use a fixed value for BER in the above equation, we get a fixed trigger time of usage errors, which results in a very restrictive model. We instead randomize usage error development: we assume that $1/\text{BER}$ is just the mean of the number of bytes accessed after the disk develops an usage error, and we assume that usage error development follows a normal distribution with mean $1/\text{BER}$ and small variance σ (e.g., 20% of the mean). We first generate a Gaussian random variable $X \sim N(1/\text{BER}, \sigma)$, and then trigger a usage error once Bytes_Written + Bytes_Read/RW_Weight $>= X$. For the read/write weight RW_Weight we use values between 1 and 9.

3. **Location of triggering error.** Assuming that a disk develops a triggering error (either age or usage) at time t_c (expressed in hours), we determine its exact location l_c on disk as a uniformly distributed random variable over all disk sectors.

4. **Number of triggered errors.** We determine the number of triggered LSE from Figure 8 in [2]. Using curve fitting in Mathematica, we determine that the probability that a disk develops x triggered errors is given (approximately) by the function $f(x) = 1.04x^{-0.185} - 0.42$.

5. **Location of triggered LSEs.** We assume that the triggered LSEs are clustered around the triggering error, with a relative distance following the piecewise uniform distribution from Figure 5 in the NetApp study.

6. **Time of triggered LSEs.** The inter-arrival time for each LSE from the previous one in the cluster is modeled with the piecewise uniform distribution from Figure 7 in the NetApp study.

We list the range of parameters used in our LSE model in Table 3.

Parameter	Range/value	Justification
Max number of errors	100	[2]
BER	$[10^{-15}, 10^{-13}]$	[6]
RW_Weight	[1,9]	Heuristic assumption
Deviation σ of usage error development	20% of mean	Heuristic assumption

Table 3: Parameter ranges in LSE model.

3.3 Model validation

We perform several experiments to validate our LSE model. We generate age-related LSEs for 100,000 disks

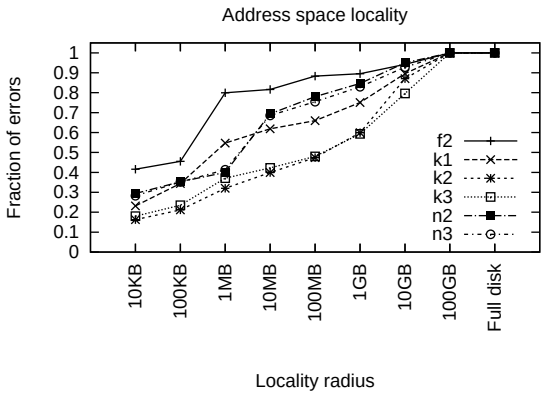

Figure 1: Fraction of errors within a given radius of an existing LSE in our simulation model.

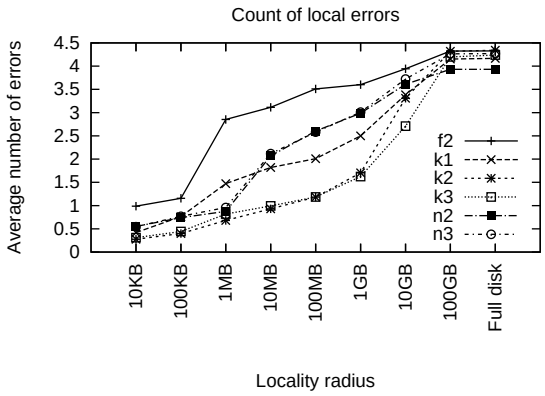

Figure 2: Average number of errors within a given radius of an existing LSE in our simulation model.

using our model and based on Figures 1, 5, 7, 8 and 10 of the NetApp study. While Figures 8 and 10 represent distributions for all disk models, Figures 1, 5 and 7 give different distributions depending on the disk model. There are six different enterprise models common to these three figures (denoted f-2, k-1, k-2, k-3, n-2 and n-3). These disk models are anonymized in the NetApp paper and we do not have information about exact disk characteristics. According to the NetApp study, drives labeled with the same letter have the same (anonymized) manufacturer, and a higher number denotes higher drive capacity (e.g., k_1, k_2 and k_3 have the same manufacturer and increasing capacities).

As monthly error rates and inter-arrival time for age errors in our simulation are generated exactly as in the NetApp study, we focus on validating our spatial LSE model. Our main goal is to validate assumptions we make due to incomplete data in the distribution of LSE location on disk, as explained above. For that, we regenerate graphs from Figures 5 and 6 in the NetApp study after the location of age errors is generated with our simulation model. Note that the results from Figure 6 are not used in our simulation model at all.

As in Figures 5 and 6 in [2], Figure 1 shows the probability of a new error arising within a given radius of an existing error, and Figure 2 shows the average number of errors within a given radius of an LSE, for the six disk models described above.

We observe that our simulation model closely reflects the results from the NetApp study. For disk models that exhibit high locality (e.g., f-2), the results of the simulation are within 1% of the study results. For models with a lower degree of locality, our simulation model slightly over-estimates the two metrics, but our simulation results differ by 6% on average from the study results.

Due to its simplicity and accuracy, we believe our LSE model is of general and practical value in the study of LSEs.

4 Scrubbing Strategies

In this section, we give the first formalization of scrubbing strategies in the literature that takes into account information about the disk model and its history. Most systems today use a simple constant-rate sequential scrubbing strategy. To capture the spatial and temporal locality of LSE development, we expand the space of scrubbing strategies across several dimensions. First, we propose a *staggered* strategy that traverses disk regions more rapidly than sequential reading. Thanks to the spatial locality of LSEs, it discovers LSEs faster than sequential scrubbing. We evaluate the performance impact of staggering, and determine parameters for which its overhead—resulting from frequent disk-head movement—is minimal (2%) compared with sequential scrubbing. Second, we consider scrubbing strategies that adaptively change their scrubbing rate according to drive age and the history of LSE development. Based on these new ideas, we propose an expanded design space of scrubbing strategies.

4.1 Formal Definition

Our formalization of scrubbing strategies accounts for disk model and age, as well as historical factors, including disk usage, the number of developed latent errors, and the scrubbing history.

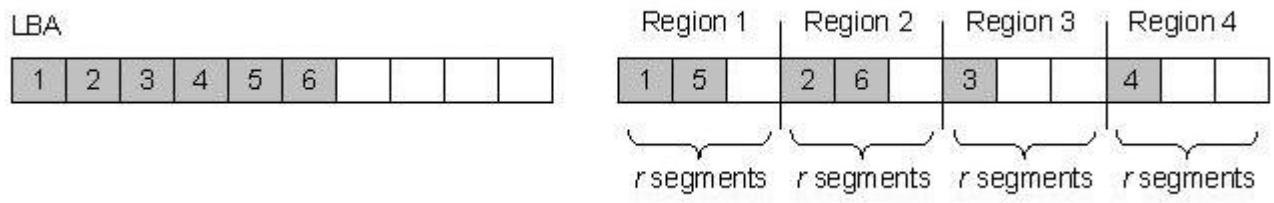

Figure 3: Representation of sequential (left) and staggered (right) scrubbing strategies.

Formally, we define a scrubbing strategy as a function of the disk age t, cumulative disk write and read usage, latent error distribution, disk failure distribution, latent error development history and scrubbing history. This function outputs the number and addresses of sectors to be scrubbed in the current time interval t.

Definition 1. A scrubbing strategy for a disk with n sectors is a function S. For inputs disk age t, cumulative disk write W_t and read usage R_t, latent error distribution $\mathcal{P}_{\mathsf{LSE}} \in \mathcal{L}$, disk failure distribution $\mathcal{P}_{\mathsf{DF}}$ in space $\mathcal{F}$, latent error development history $L_t^h = \{E_1, \ldots, E_{t-1}\}$ (as defined in Section 3.2), and scrubbing history $S_t^h = \{v_i, [1,n]^{v_i}\}_{i=1,\ldots t-1}$ (including the number and addresses of sectors scrubbed at all previous time intervals), it outputs the number of sectors selected for scrubbing v_t, and their logical block addresses $(\mathsf{LBA}_1, \ldots, \mathsf{LBA}_{v_t})$.

For example, assuming that LBAs are between 0 and $n-1$, the sequential strategy with constant-rate r can be formally defined as $S(t, W_t, R_t, \mathcal{P}_{\mathsf{LSE}}, \mathcal{P}_{\mathsf{DF}}, L_t^h, S_t^h) = \{r, (rt+1 \bmod n, \ldots, r(t+1) \bmod n)\}$. Note that the constant-rate sequential strategy only depends on disk age, but it does not take into account other disk characteristics or history of error development.

We leave the definition of the disk failure distribution as general as possible. It can depend on disk age, disk usage and failure history, similar to the definition of LSE distribution. We omit the disk failure history from the scrubbing strategy definition since once a disk fails, it is replaced with a new one and our model is restarted.

4.2 Staggered scrubbing

Our staggered scrubbing regime—again, aimed at exploiting the spatial locality of LSEs—is as follows. The disk is partitioned into m regions, each consisting of r segments. Staggered scrubbing reads the first segment of each disk region in turn, ordered by LBA. Then it reads the second segment in each disk region, and so forth, up to the r^{th} segment, as depicted in Figure 3. (Once a full scrubbing pass is complete, it is initiated again with the first segment.)

Intuitively, staggering is effective because LSEs tend to arise in clusters: if a given region develops LSEs, there is a good chance that many of its segments will contain at least one. Consequently, repeated sampling of a region—which is what staggering accomplishes over a full scrubbing pass—is more effective than full sequential scrubbing of a region. To see this more clearly, consider an extreme case of clustering: suppose that when a region develops an LSE, all of its segments develop one. In this case, sampling any one segment suffices to detect an LSE-affected region; there is no benefit to scrubbing more than one segment per region. So it is best to sample one segment per region, move on as quickly as possible, and return later to check for fresh LSEs, i.e., to stagger.

Staggering does have a drawback, though. It requires more disk-head movement than sequential scrubbing. (Sequential scrubbing is clearly optimal in terms of disk-head movement.) Thankfully, as we show next, for carefully chosen parameters, the slowdown due to disk-head movement in staggered scrubbing is minimal.

We determined through experiments parameters for the staggered strategy that do not affect performance. The first question we needed to answer is the optimal request size when reading from disk sequentially. As suggested by previous literature [12], read performance improves with increasing request sizes, as function calls and interrupts introduce a performance penalty.

We performed a first experiment in which we read 16GB from a 7200 RPM Hitachi drive using request sizes between 1KB and 64KB. We found that a disk request size of 16KB is nearly optimal; performance improves negligibly for larger request sizes. This suggets that re-

quest sizes in sequential scrubbing strategies should be at least 16KB.

Second, we want to quantify the performance overhead for staggered scrubbing versus sequential reading from disk. We consider staggered scrubbing with regions of different sizes, ranging from 50MB to 500MB, and different request sizes, ranging from 32KB to 2MB. We found out that, while the overhead of staggering for small request sizes (32KB or 64KB) is large (a factor of 5 to 8), the overhead becomes minimal when the request size increases to several MB. For instance, for a request size of 1MB or 2MB, the overhead is about 2%.

These experimental findings provide guidance for our parameter choices in staggered scrubbing. To minimize the performance impact of staggering, we choose a segment size of 1MB. For that segment size, our results show that the staggering overhead is not highly dependent on the region size. We thus choose a region size that aligns with the radius of most error clusters (128MB).

4.3 Strategies with Adaptive Scrubbing Rates

To capture temporal locality of latent sector errors, we introduce scrubbing strategies with scrubbing rates that change adaptively according to drive history. From the results in the NetApp study, we know that monthly LSE rates are fairly constant before the development of the first LSE in a drive. (Again, an exception is the first 60 days of drive operation, which exhibit slightly lower LSE rates.) Once a first LSE develops, i.e., a triggering error, more errors are likely to develop shortly afterward.

We propose to start with a scrubbing rate SR_First60 in the first 60 days of disk operation, and change it to rate SR_PreLSE before any LSEs are detected. Once the disk develops a first LSE, the strategy enters into an *accelerated interval* (with length Int_Acc) and adjusts the scrubbing rate to SR_Acc. At the end of the accelerated interval, the scrubbing rate is modified to SR_PostLSE. The process is repeated every time a LSE is detected: the strategy enters an accelerated interval with an adjusted scrubbing rate, and then reverts to SR_PostLSE. Disks that never develop an LSE are scrubbed with rate SR_First60 in the first 60 days of operation and SR_PreLSE after that.

4.4 Modeling the Design / Search Space of Scrubbing Strategies

Combining the ideas of staggering and adaptive scrubbing rates, we propose an expanded design space of scrubbing strategies that we will search for optimal strategies in the next section of the paper. A strategy in this design space operates as follows. Before the detection of the first LSE, the strategy proceeds in a staggered fashion with scrubbing rates SR_First60 in the first 60 days of drive operation and SR_PreLSE after that. Once a first LSE is detected, the strategy enters into an accelerated interval and switches to a sequential strategy with scrubbing rate SR_Acc. It scrubs sequentially regions of the disk centered at the detected error and continues with regions further away. When the accelerated interval ends, the strategy reverts to staggered scrubbing with rate SR_PostLSE, starting from the first disk sector.

The parameters that characterize our design space are graphically depicted in Figure 4. A point in our design space is given by coordinates (SR_First60, SR_PreLSE, SR_Acc, SR_PostLSE, Int_Acc).

To convert our design space into a search space, i.e., to specify the constraints on our search for optimal strategies, we must choose concrete parameter ranges and granularities. While this is a somewhat heuristic process, experimental guidance motivates the following choices:

- The staggered strategy uses a region of size 128MB, and a segment size of 1MB. These choices were explained in Section 4.2.

- We specify the scrubbing rates in terms of gigabytes scrubbed per hour. We constrain these rates to an interval whose maximum value corresponds to a full disk scrub in one day (which amounts to 20GB/hour for a 500GB disk). We define the search space for these scrubbing rates with a granularity of 0.5GB/hour, starting from the minimum value of 0.5GB/hour.

- The length of interval Int_Acc is a parameter with minimum value 3 hours and maximum value the time it takes to scrub the full disk sequentially with rate SR_Acc. We search this interval at a granularity of 3 hours.

- The size of the regions scrubbed sequentially in accelerated intervals is 128MB, since this is the clustering radius of about 80% of LSEs. We scrub the regions of size 128MB centered at the first error found, and then continue with the regions further away.

5 Simulation Model and Evaluation

Before describing our simulation model, we specify our new metric MLET (Mean Latent Error Time). Intuitively, for a single disk with a specified latent error model and scrubbing strategy, MLET measures the average (over LSE patterns) fraction of the total drive operation time during which the drive has undetected LSEs and is thus susceptible to data loss.

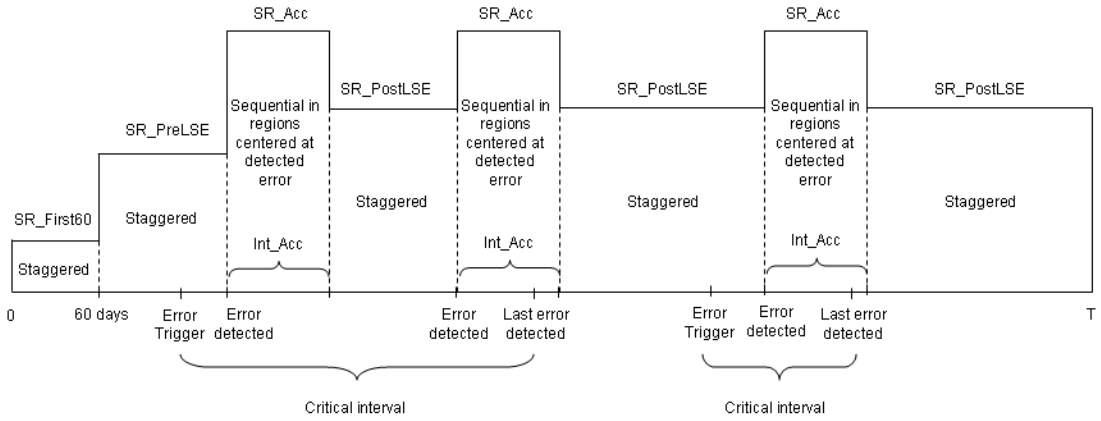

Figure 4: Search space of scrubbing strategies given by parameters SR_First60, SR_PreLSE, SR_Acc, SR_PostLSE and Int_Acc.

Formally, consider a latent sector error probability distribution $\mathcal{P}_{\mathsf{LSE}}$ from space $\mathcal{L}$ and a scrubbing strategy S from space $\mathcal{S}$. For a given pattern of latent-error development LSE from $\mathcal{P}_{\mathsf{LSE}}$, we define the Latent Error Time $\mathrm{LET}(t, \mathsf{LSE}, \mathsf{S})$ as the fraction of the time intervals up to disk age t during which the drive has undetected LSEs. $\mathrm{MLET}(t, \mathsf{S})$ is then defined as the mean of $\mathrm{LET}(t, \mathsf{LSE}, \mathsf{S})$ over the probability distribution $\mathcal{P}_{\mathsf{LSE}}$.

We note that this definition holds for a deterministic scrubbing strategy S. We could extend the definition for probabilistic strategies, to average over the scrubbing strategy distribution $\mathcal{S}$.

5.1 Simulation Model

We have written an event-driven simulation model in Java that simulates the behavior of a disk for T time intervals, each of length one hour. In our experiments, we run our simulation for maximum 24 months for 100,000 disks. (The NetApp data span 24 months of disk operation.) We consider enterprise disk model n-2 and simulate hard drives with a capacity of 500GB. We model the disk normal workload using the HP Cello 99 traces, available from the SNIA IOTTA repository [1]. In our simulation we are interested only in total number of bytes read and written per time interval (i.e., hour). We compute the number of bytes accessed for one hard drive in the original Cello traces. Since these traces are ten years old, we expect that the utilization level is low compared to today's environments. To simulate different utilization levels we scale the number of bytes accessed by a factor between 1 and 100. We simulate both sequential strategies with fixed scrubbing rates and staggered strategies with fixed and adaptive rates.

The events of interest to our simulator are the triggering of age and usage errors, detection of errors, and the moments in time when the scrubbing rate changes, i.e., the disk age reaches 60 days, an accelerated interval begins, or an accelerated interval ends. Age errors are triggered by the distribution derived from the NetApp paper, as described in Section 3.2. The simulator keeps track of the usage rates due to both normal accesses and disk scrubbing and triggers a usage error once the usage for a disk exceeds a random variable normally distributed, as described in Section 3.2.

One important challenge arises in the construction of an efficient simulator. Recall that in our LSE model, a triggering LSE is followed by a cascade of other LSEs. The interval of time between the first error trigger and the detection of all errors in a cluster is what we call a *critical interval*, depicted in Figure 4. It is possible that while in the critical interval of one cluster of errors, another cluster of errors develops. Accommodating a potentially large number of overlapping and nested critical intervals would complicate our model and simulation considerably. For this reason, we make the simplifying assumption that clusters of usage errors do not overlap. We do, however, treat the case in which an age error cluster overlaps with an usage error cluster.

In practice, following a LSE detection, a logical-to-physical remapping of the affected sector takes place. We do not consider the effect of this remapping in our simu-

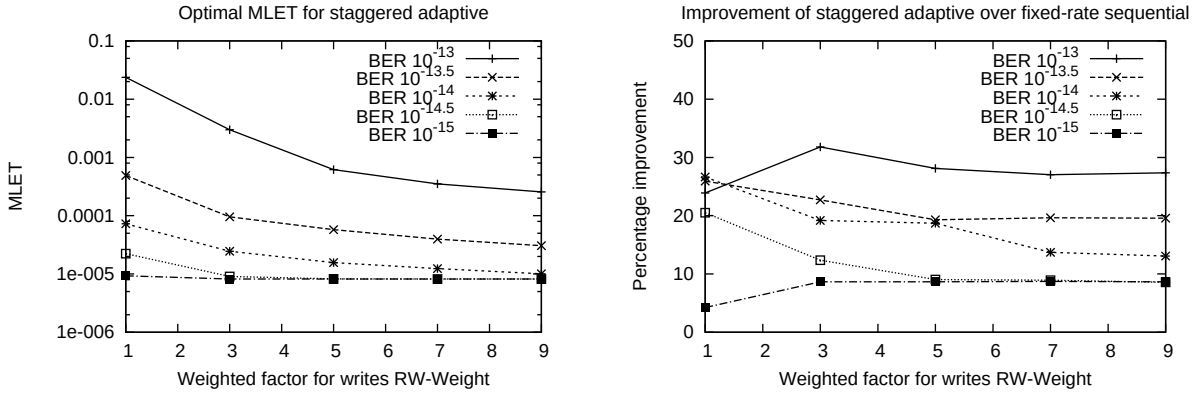

Figure 5: Optimal MLET for staggered adaptive strategies (left) and its relative percentage improvement compared to optimal fixed rate sequential strategies (right) as a function of different BERs and weighted factors for write.

lation model, but this needs to be addressed in an actual implementation of scrubbing strategies in hard drives.

5.2 Simulation Results

Our goal is to determine optimal scrubbing strategies in the design space outlined in Section 4.4. Since our design space for scrubbing strategies proved to be too large to be searched exhaustively in an efficient manner, we implemented a more efficient heuristic search algorithm. Based on brief experimentation, we believe that this heuristic finds strategies close to optimal. For a fixed BER, read/write weight RW_Weight, and disk workload, the algorithm to determine an approximation to the optimal scrubbing strategy in our design space is the following:

- We search exhaustively for the scrub rate λ (between 0.5GB/hour and maximum scrubbing rate) that achieves the minimum MLET for staggered fixed-rate strategies.

- We vary the rate in the accelerated interval between λ and the maximum scrub rate (given by a full scrub per day), and the length of the accelerated interval (between 3 hours and the time it takes to scrub the full disk with the accelerated scrub rate). We determine thus the scrub rate λ_{acc} and the length of accelerated interval $\mathsf{int\_acc}$ that minimize MLET.

- We vary the rate in the first 60 days from 0.5GB/hour to the maximum allowed scrub rate, and determine λ_{60} that minimizes MLET. Similarly, we vary $\mathsf{SR\_PreLSE}$ and $\mathsf{SR\_PostLSE}$ to determine $\lambda_{\mathsf{prelse}}$ and $\lambda_{\mathsf{postlse}}$.

- We output the point $(\lambda_{60}, \lambda_{\mathsf{prelse}}, \lambda_{\mathsf{acc}}, \lambda_{\mathsf{postlse}}, \mathsf{int\_acc})$ as an estimate of the optimal strategy.

In the rest of the paper, we sometimes refer to the output of the previous algorithm as "optimal strategy".

Optimal strategy dependence on different BER and read/write weights. First, we show how the optimal scrubbing strategy depends on the drive BER and the read/write weight RW_Weight. We plot on the left graph in Figure 5 the optimal MLET for staggered adaptive strategies and on the right graph in Figure 5 its relative improvement compared to optimal fixed-rate sequential strategies. We vary BER between 10^{-15} and 10^{-13}, and the read/write weight between 1 (i.e., read and write contribute equally to disk wear-out) and 9 (i.e., contribution of reads to disk wear-out is 9 times lower than that of writes).

The left graph in Figure 5 shows how MLET decreases for more reliable disks (i.e., disks with higher BER): for instance, for a read/write weight of 1, MLET varies between 0.031 for a 10^{-13} BER to $9.69 \cdot 10^{-5}$ for a 10^{-15} BER. As expected, MLET also decreases when the disk wear-out due to reads is lower (i.e., the read/write weight increases), as the disk is developing fewer usage errors.

From the right graph in Figure 5, we infer that the staggered adaptive strategy improves MLET relative to the optimal fixed-rate sequential strategy by at most 30%. Improvements are larger for disks with higher development of usage errors. We expect that this effect will be amplified when considering RAID-5 or RAID-6 configurations with multiple disks. In RAID-5, for instance, data loss occurs when a drive failure is coupled with a latent error on any of the other drives. The vulnerability interval due to latent errors (the time intervals in which at least one drive has undetected LSEs) consists of all vulnerability intervals of the drives in the RAID configuration. Consequently, a reduction in the MLET metric for one drive will produce an amplified reduction on the length of the vulnerability interval for the array (roughly

BER		Weighted factor for writes				
		1	3	5	7	9
10^{-13}	fixed-rate	4	0.5	0.5	0.5	1
	adaptive	(0.5,10,18.5,2.5)	(0.5,12.5,14.5,0.5)	(0.5,0.5,12.5,0.5)	(0.5,0.5,18.5,0.5)	(0.5,1,17,1.5)
$10^{-13.5}$	fixed-rate	0.5	1.5	2.5	4	5
	adaptive	(0.5,0.5,12.5,0.5)	(1,1.5,15.5,1.5)	(3,3,14,3)	(3.5,3.5,17,3.5)	(5,5,18,5)
10^{-14}	fixed-rate	2	6	9.5	12.5	17.5
	adaptive	(1,2,18,1)	(2,6,19.5,5)	(10,10,18.5,10)	(13,13,19,13)	(18,18,19.5,18)
$10^{-14.5}$	fixed-rate	6.5	19	20	20	20
	adaptive	(7,7,19,7)	(12.5,20,20,20)	(17,20,20,20)	(17,20,20,20)	(17,20,20,20)
10^{-15}	fixed-rate	20	20	20	20	20
	adaptive	(19,19,19,19)	(17,20,20,20)	(17,20,20,20)	(17,20,20,20)	(17,20,20,20)

Table 4: Optimal points for sequential fixed-rate and adaptive staggered strategies for different BERs and weighted factors for writes. For sequential fixed-rate strategy, the table includes the optimal scrubbing rate. For the adaptive staggered strategy, the table shows the optimal point (SR_First60, SR_PreLSE, SR_Acc, SR_PostLSE).

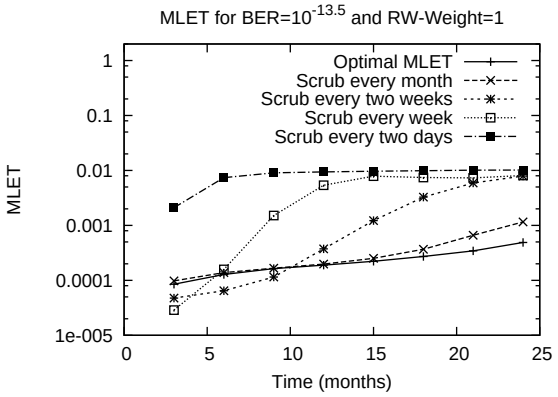

Figure 6: MLET for optimal strategy and several sequential strategies for disks with high usage errors.

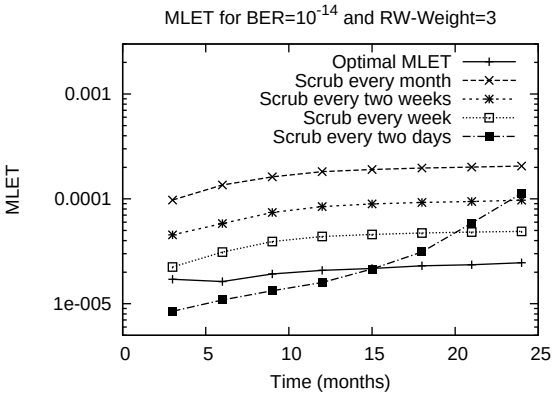

Figure 7: MLET for optimal strategy and several sequential strategies for disks with medium usage errors.

scaled by the number of drives in the RAID configuration).

Table 4 gives an interesting insight on the optimal scrubbing rates used by both fixed-rate sequential and adaptive staggered strategies. For disks featuring high development of usage errors (due to high BER, and low read/write weight), the optimal fixed-rate sequential strategy is using a fairly low scrubbing rate (since in this case the scrubbing process itself will contribute to disk wear-out and LSE development). The optimal staggered adaptive strategy also uses low scrub rates, except for accelerated intervals, when the scrubbing rate is increased to almost maximum allowed rate to detect LSEs quickly. In contrast, for disks developing few usage errors (due to low BER and high read/write weight), the optimal scrubbing strategies (both sequential and staggered adaptive) use a high scrubbing rate that is close to the maximum allowed rate.

Improvement of staggered adaptive strategy over several widely used fixed-rate sequential scrubbing strategies. We compare next the MLET metric for the optimal adaptive staggered strategy and various fixed-rate sequential strategies (i.e., scrub the disk once a month, once every two weeks, once every week, and once every two days). These fixed-rate sequential strategies are widely used today in many systems. Graphs in Figures 6, 7 and 8 show the MLET metric for these strategies as a function of the simulation interval. The results demonstrate that by using more intelligent scrubbing than the ad-hoc approaches in use today, the MLET metric can be improved by at least a factor of two and at most a factor of 20.

An important observation derived from these graphs is that optimal strategies are highly dependent on disk characteristics. For disks that develop a high number of usage errors (Figure 6 with BER $10^{-13.5}$ and the read/write weight 1), the optimal adaptive staggered strategy is clos-

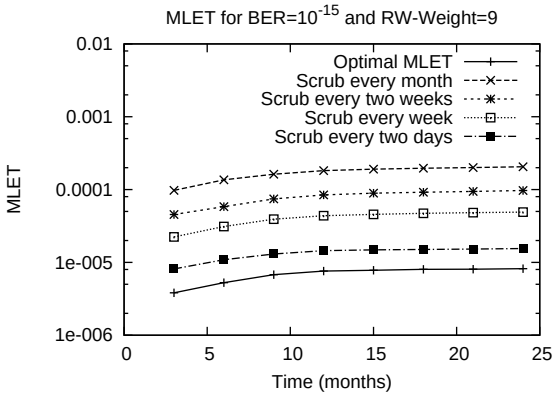

Figure 8: MLET for optimal strategy and several sequential strategies for disks with low usage errors.

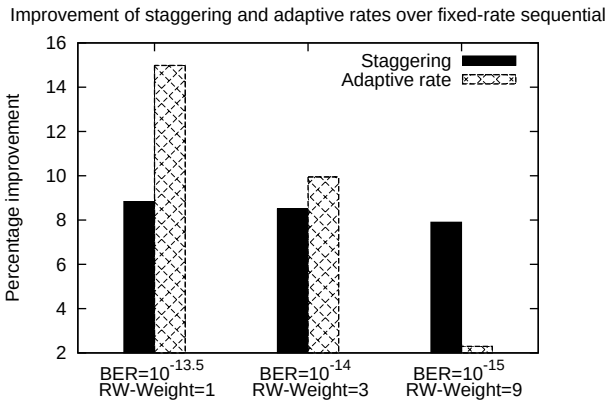

Figure 9: Relative improvement in MLET for staggering and adaptive rates compared to fixed-rate sequential.

est to scrubbing the disk once every month (i.e., infrequent scrubbing). For disks with medium number of usage errors (Figure 7 with BER 10^{-14} and the read/write weight 3), the optimal strategy is closer to scrubbing the disk once every week. In Figure 8, disks that develop low number of usage errors (e.g., BER 10^{-15} and the read/write weight 9) have optimal strategies closer to scrubbing every two days. This clearly demonstrates that it is infeasible to develop a good "one-size-fit-all" recipe for disk scrubbing.

Interestingly, Figures 6 and 7 show that the optimal strategy for time t is not always the optimal strategy for all previous time intervals. This observation suggests that we could achieve further optimizations when designing scrubbing strategies by expanding our search space. In particular, an idea that deserves further exploration is to periodically adapt the scrubbing strategy over time. Instead of computing one optimal strategy for the entire drive operational time, we could instead compute new optimal strategies for short time intervals (e.g., 3 or 6 months). With this approach, the optimal strategy for disks that develop a medium number of errors, for instance, is to scrub with a constant rate (once every two weeks) for the first 15 months, and then switch to an adaptive staggered strategy.

Benefit of staggered and adaptive strategies. We assess next the benefit of our two main optimizations: using a staggered approach for scrubbing, and varying scrubbing rates adaptively. We show in Figure 9 relative improvements of these two optimizations compared to the optimal fixed-rate sequential strategy. We plot results for disks with three different characteristics, classified by the occurrence of high, medium or low occurrence of usage errors, respectively.

We observe that the idea of staggering compared to sequentially reading the disk produces a steady improvement in MLET by around 10% for all disk characteristics. On the other hand, adaptively changing the scrubbing rate has a greater impact on disks that develop a higher number of usage errors. The relative improvement in MLET by adaptively changing the scrubbing rate is as high as 15% for disks with a high number of usage errors, and as low as 2% for most reliable disks. These results are consistent with our previous observation that the optimal scrubbing strategy for disks with few usage errors is scrubbing at the maximum fixed rate.

Interestingly, a paper concurrently and independently written [13] shows that our experimental results might underestimate the benefit of the staggering technique. Schroeder et al. [13] evaluate staggered scrubbing in comparison with fixed-rate sequential strategies on real failure data and report that staggered scrubbing can improve mean time of error detection compared to sequential scrubbing by up to 40%. While Schroeder et al. use a different metric in comparing different scrubbing strategies, these results confirm the benefit of staggering.

Optimal strategy dependence on disk workloads. Finally, we assess the impact of different disk workloads on optimal scrubbing strategies. We consider the workloads of one disk from the HP Cello 1999 I/O traces, and scale them by a factor of 1, 10 and 100. We plot on the left of Figure 10 the MLET value for optimal staggered adaptive strategy and on the right its relative improvement compared to fixed-rate sequential strategies.

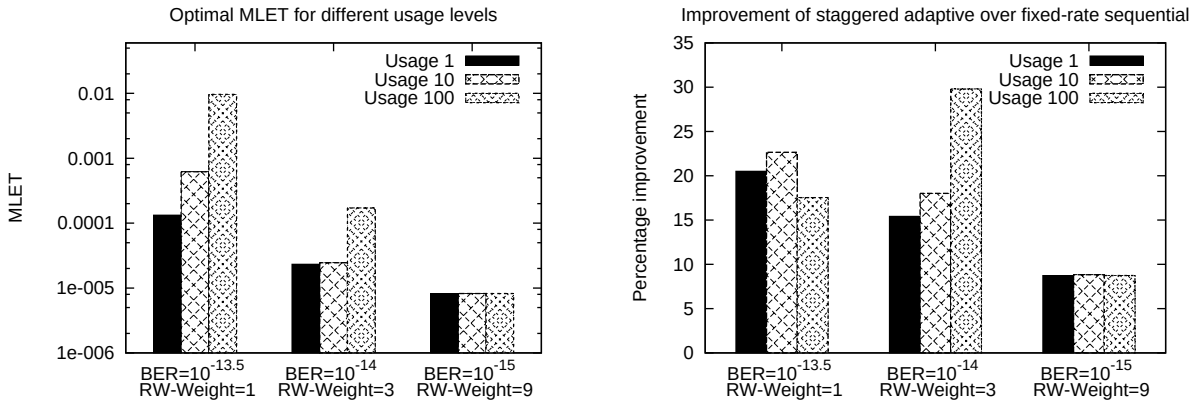

Figure 10: Optimal MLET for staggered adaptive strategies (left) and its relative percentage improvement compared to optimal fixed rate sequential strategies (right) for different disk characteristics and different workloads.

In both graphs, usage levels are scaled by a factor of 1, 10 and 100, respectively. As in previous experiments, we consider disks that develop a high, medium and low level of usage errors.

The left graph in Figure 10 shows that disks developing high and medium number of usage errors exhibit sensitivity to normal access workloads. In particular, scaling the disk workloads by a factor of 10 has the effect of increasing the optimal MLET metric by an order of magnitude for disks developing a high number of usage errors. Disks that exhibit low number of usage errors are not sensitive to disk workloads at all.

The right graph in Figure 10 shows the relative improvement of the optimal staggered adaptive strategy compared to the optimal fixed-rate sequential strategy for different disk usage levels. Disks exhibiting high and medium development of usage errors benefit mostly from the staggered adaptive technique. For these types of disks, the relative improvements of the staggered adaptive strategy increase with higher disk utilization. The exception is the case of disks developing high number of usage errors under heavy workload (scaled by a factor of 100). In that case, we conjecture that the number of usage errors increases greatly, leading to lower relative improvements of the staggered adaptive strategy than for lower disk utilization. We observe again that disks developing a low number of errors are insensitive to disk workloads: the relative improvement of the staggered adaptive strategy is around 10%, independent of the disk workload.

Discussion. We have demonstrated that we can design more intelligent scrubbing algorithms than those in use today by taking into account disk characteristics and the history of error development. We have characterized the resilience of a single drive to latent sector errors by defining the new MLET metric. Our results demonstrate that optimal scrubbing strategies need to be carefully crafted for different disk characteristics. In particular, optimal strategies are highly dependent on the BER and the read/write weight RW_Weight of a disk.

For disks that develop a high number of usage errors, scrubbing benefits greatly from adaptively changing rates. The optimal strategy uses a low scrubbing rate, that is increased to almost the maximum allowed rate in the accelerated interval immediately following the detection of a LSE. For disks that develop a low number of usage errors, the optimal strategy uses the maximum allowed scrubbing rate that does not interfere with the normal disk usage. Staggering across disk regions instead of sequentially reading the disk improves the MLET metric for all disk models.

Our optimal scrubbing strategies can improve the MLET metric compared to widely used strategies (e.g., scrub the disk sequentially once every week) by an order of magnitude. We expect that this effect will be amplified when considering the MTTDL metric for an array of disks (e.g., RAID-5 or RAID-6 configuration).

A limitation of the current work is the high sensitivity of the results to disk parameters that are not always made public by disk manufacturers. We hope that, as more failure data becomes available, our results can be further refined by the community.

6 Conclusions

Our work is a first step in the exploration of more intelligent scrubbing strategies for hard drives. It shows that single drive reliability can be greatly improved by expanding the design space for scrubbing strategies beyond naïve sequential and constant-rate approaches.

Several challenging options for further research arise in our work. The first is an expansion of our design and search spaces for scrubbing strategies. Appealing to search heuristics such as hillclimbing or simulated annealing would enable us to consider a more fine-grained and sophisticated design space.

Second, we plan to evaluate the performance overhead of various scrubbing strategies in conjunction with realistic disk workloads.

Third, with the emergence of FLASH technology, an intriguing question is how (and if) our results translate into the FLASH realm. With completely different physical characteristics than hard drives, and a complex physical-to-logical translation layer, FLASH would seem a challenging target for the development of latent error and scrubbing models.

Finally, we have only studied the effect of scrubbing on single-drive reliability. Extension of our work to a systemic analysis in the context of replication systems like RAID seems an interesting area of future research.

7 Acknowledgements

We thank Ron Rivest, Burt Kaliski and Kevin Bowers for numerous insightful discussions during the progress of this work. We also thank Bianca Schroeder and Georgios Amvrosiadis for conversations on latent error modeling and staggering strategies. Finally, we would like to extend our gratitude to our shepherd, Jim Plank, and the anonymous reviewers for their careful suggestions in revising the final version of the paper.

References

[1] The SNIA IOTTA Repository. `http://iotta.snia.org/`.

[2] L.N. Bairavasundaram, G.R. Goodson, S. Pasupathy, and J. Schindler. An analysis of latent sector errors in disk drives. In *ACM SIGMETRICS*, pages 289—300, 2007.

[3] M. Baker, M. A. Shah, D. S. H. Rosenthal, M. Roussopoulos, P. Maniatis, T. J. Giuli, and P. P. Bungale. A fresh look at the reliability of long-term digital storage. In *1st ACM SIGOPS/EuroSys*, pages 221—234, 2006.

[4] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. Raid: high-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185.

[5] A. Dholakia, E. Eleftheriou, X. Hu, I. Iliadis, J Menon, and K. K. Rao. A new intra-disk redundancy scheme for high-reliability RAID storage systems in the presence of unrecoverable errors. *ACM Transactions on Storage*, 4(1), 2008.

[6] J.G. Elerath and M. Pecht. Enhanced reliability modeling of RAID storage systems. In *37th Annual IEEE/IFIP DSN*, pages 175—184, 2007.

[7] I. Iliadis, R. Haas, X. Y. Hu, and E. Eleftheriou. Disk scrubbing versus intra-disk redundancy for high-reliability RAID storage systems. In *ACM SIGMETRICS*, pages 241—252, 2008.

[8] H. Kari. *Latent Sector Faults and Reliability of Disk Arrays*. PhD thesis, Helsinki University of Technology, 1997.

[9] N. Mi, A. Riska, E. Smirni, and E. Riedel. Enhancing data availability through background activities. In *38th Annual IEEE/IFIP DSN*, 2008.

[10] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID). In *ACM SIGMOD*, pages 109—116, 1988.

[11] E. Pinheiro, W. D. Weber, and L. A. Barroso. Failure trends in a large disk drive population. In *5th USENIX FAST*, 2007.

[12] E. Riedel, C. van Ingen, and J. Gray. A performance study of sequential I/O on windows NT. In *Second USENIX Windows NT Symposium*, 1998.

[13] B. Schroeder, S. Damouras, and P. Gill. Understanding latent sector errors and how to protect against them. In *8th USENIX FAST*, 2010.

[14] B. Schroeder and G. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean too you? In *5th USENIX FAST*, 2007.

[15] T. Schwarz, Q. Xin, E. L. Miller, D. D. E. Long, A. Hospodor, and S. Ng. Disk scrubbing in large archival storage systems. In *IEEE 12th MASCOTS*, 2004.